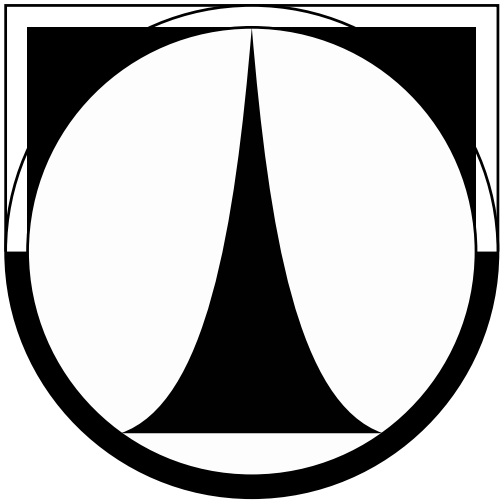
Technická univerzita v Liberci

Fakulta mechatroniky, informatiky a mezioborových studií

BAKALÁŘSKÁ PRÁCE



Zdeněk Perutka

Implementace DNSSEC do síťového protokolu Verse

Implementation of DNSSEC to Verse network protocol

Ústav Nových technologií a aplikované informatiky

Vedoucí bakalářské práce: Ing. Jiří Hnídek

Studijní program: **Informační technologie**

Studijní obor: **Informační technologie**

Liberec 2011

Zadání

Poděkování

Na tomto místě bych chtěl poděkovat zejména své rodině a přátelům za podporu, sestře Šárce za trpělivou pomoc s pravopisem. Velký dík patří také vedoucímu práce, Ing. Jiřímu Hnídkovi, za konzultace a užitečné rady při řešení zadání.

Kontakt

E-mail: zdnek.perutka@tul.cz

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 o právu autorském, zejména § 60 (školní dílo).

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce a konzultantem.

Datum

Podpis

Anotace

Tato bakalářská práce se zabývá rozšířením síťového protokolu Verse, zabývajícím se real-timovým sdílením dat, o podporu DNSSEC (zabezpečeným DNS). Toto rozšíření Verse protokolu je důležité pro zvýšení bezpečnosti Verse protokolu.

V úvodu práce je stručně popsána podstata fungování DNSSEC. Následně je předložen stručný přehled knihoven umožňujících DNSSEC a výběr nejvhodnější knihovny pro implementaci do knihovny implementující protokol Verse.

Jedním z hlavních účelů práce bylo vytvoření dostatečně komplexní callback funkce, která by byla schopna předávat programátorovi Verse klienta bezpečnostní informace o DNSSEC a certifikátech. Bylo tedy nutné aby funkce byla dostatečně univerzální. Práce se také zabývá vhodným rozšířením Verse API, tak aby bylo možné nové funkce používat.

Funkčnost a správnost rozšířeného Verse protokolu je v závěrečné části práce vyzkoušena na vzorové aplikaci, která byla nově rozšířena za účelem testování nových funkcí. Jelikož se DNSSEC zatím v praxi příliš nepoužívá a není mnoho „podepsaných“ domén, bylo testování provedeno připojením pouze ke dvěma podepsaným doménám.

Klíčová slova: DNSSEC, callback funkce, API, ldns, Verse protokol

Annotation

The thesis is focused on extension of Verse network protocol which is used for real-time data sharing. The extension should base on including a support for DNSSEC (secured DNS) which is very important in context of security enhancement.

Introduction of the thesis briefly describes the nature of the DNSSEC. Latter a brief overview of libraries which enable DNSSEC is offered following by a selection of the best library to be used for implementation into the Verse protocol library.

One of the main purposes of the study was to create a callback function complex enough which would be able to submit security information about DNSSEC and certificates to a programmer of Verse client, keeping in mind that the function has to be sufficiently universal. The propriate extension of Verse API is also included so that it would be possible to use the new functions.

Functionality of the extended Verse protocol was checked on a sample application in the final parts of the thesis. A sample application already written was also extended in order to test new functions. Testing was proceeded by connecting to two signed and one unsigned domains.

Key words: DNSSEC, callback function, API, ldns, Verse protocol

Obsah

Seznam symbolů a zkratek

AD Authenticated Data

API Application Programming Interface

BSD Berkeley Software Distribution

DNS Domain Name System

DNSSEC Domain Name System Security Extension

EPP Extensible Provisioning Protocol

ID jedinečný identifikátor

IP Internet Protocol

IPv4 Internet Protocol version 4

IPv6 Internet Protocol version 6

NSEC Next Secure

NSEC3 Next Secure 3

RA Recursion Available

RD Recursion Desired

RR Resource Record

SVN Subversion

TC Truncated

TCP Transmission Control Protocol

TLS Transport Layer Security

TSIG Transaction Signature

TTL Time To Live

UDP User Datagram Protocol

1 Úvod

Úvodem bude stručně popsán Verse protokol a vysvětlena podstata práce DNSSEC. Dále budou vysvětleny i další pojmy, týkající se práce na rozšíření Verse protokolu.

1.1 Verse protokol

Verse je síťový protokol typu klient-server. Je určen pro real-timové sdílení dat mezi grafickými aplikacemi.

Původní Verse protokol byl vyvíjen Uni-Verse konsorciem v rámci 6. rámcového programu Evropské unie. Do konsorcia patřilo několik významných univerzit a výzkumných institucí jako KTH, Fraunhoufer Institut, Helsinky University of Technology, Interactive Institute a Blender Foundation. Verse protokol byl od počátku navrhován pro efektivní real-timové sdílení dat, především mezi grafickými aplikacemi a měl ambici stát se univerzálním protokolem pro komunikaci mezi grafickými aplikacemi. Po ukončení financování od Evropské unie vývoj protokolu ustal a ten se bohužel z mnoha důvodů nakonec nerozšířil [1].

V současné době vyvíjí Ing. Jiří Hnídek novou verzi Verse protokolu, která má být robustnější a především snadněji implementovatelná do současných i budoucích aplikací. Protokol je implementován v jazyce C.

1.2 DNSSEC

DNSSEC [2] je bezpečnostním rozšířením protokolu DNS [3], používaného pro překlad doménových jmen na IP [4] adresy. DNSSEC zajišťuje, že informace o IP adrese nebyla podvržena, tedy že nebyla poskytnuta jiná IP adresa, než je skutečná adresa serveru. Při návrhu DNSSEC byl kladen maximální důraz na zpětnou kompatibilitu, protože díky tomu jsou aplikace implementující DNSSEC schopny komunikovat s aplikacemi používající pouze původní DNS.

1.2.1 Princip DNS

Jedná se o hierarchický systém. Počítač má nastavenu IP adresu DNS serveru, resovleru, který provozuje typicky poskytovatel internetu. Pokud je zadáno doménové jméno nějakého serveru (např. www.tul.cz), se kterým má být komunikováno, počítač pošle dotaz na resolver. Tím je zahájeno rekurzivní dotazování. DNS server se spojí s kořenovým (root) serverem, který sice informaci o konkrétních jménech nemá, ale jako odpověď pošle IP adresu DNS serveru pro doménu cz. Který odešle IP adresu *name serveru* tul.cz a ten už odešle IP adresu webového serveru www.tul.cz. Tuto informaci pak DNS server přepošle počítači, ze kterého byl odeslán dotaz.

Především během komunikace mezi resolverem a jednotlivými name servery, vzniká šance na podvržení DNS informace [5].

DNS servery si nalezené IP adresy ukládají do cache, tak aby při opětovném dotazu na stejnou doménu nebylo nutné znovu se rekurzivně dotazovat. Tazateli je pak odeslán záznam uložený v lokální cache serveru. To je možné díky poli TTL, které značí jak dlouho je možné DNS záznam používat. Pole TTL je obsaženo v každém záznamu [6]. Ovšem pokud je v hlavičce dotazu nastaven příznak RD, znamená to, že tazatel požaduje rekurzivní doptávání. Server pak do odpovědi nastaví příznak RA značící, že je ochotný rekurzivní doptávání provést a provede jej [7].

Stejným způsobem jako server cachuje DNS záznamy i počítač, ze kterého je posílán dotaz. V Linuxu je možné cache vyprázdnit restartováním démona Nscd, starajícího se o správu DNS cache. Stačí zadat */etc/init.d/nscd restart*.

Komunikace DNS probíhá skrz port 53 transportních protokolů UDP [8] a TCP [9]. Díky jednoduchosti protokolu UDP je běžný dotaz posílán pomocí tohoto protokolu stejně jako odpověď. Pokud je odpověď delší než 512 Bytů, odešle se pouze částečná odpověď s příznakem TC v hlavičce. Tento příznak signalizuje, že se nejedná o kompletní odpověď. V případě že tazatel potřebuje celou odpověď, pošle stejný dotaz tentokrát skrze TCP protokol a je mu pomocí stejného protokolu poslána kompletní odpověď [10].

1.2.2 Princip DNSSEC

DNSSEC využívá digitální podpis [11], který je v dnešní době běžným prostředkem pro ověření autenticity informací. Pro doménu, která má být podepsána, je vygenerován privátní a k němu náležící veřejný klíč, funguje zde tedy princip asymetrické šifry, kdy data zašifrovaná privátním klíčem lze dešifrovat veřejným klíčem. Privátní klíč je nutné držet v tajnosti a jsou jím podepisovány DNS informace. Zatímco pomocí veřejné klíče je ověřována pravost těchto podpisů. Veřejný klíč je nahrán do zónového souboru nadřazené domény [12].

Veřejné klíče jsou pak podepisovány stejným způsobem. Tedy v nadřazené doméně je opět vygenerován privátní a veřejný klíč. Privátním klíčem jsou podepsány veřejné klíče zóny a veřejný klíč, který je opět uložen o úroveň výš a slouží k ověření pravosti podpisu.

Takto se postupuje až ke kořenovému serveru. V praxi by tak mělo stačit mít důvěryhodný veřejný klíč ke kořenové doméně, s jehož pomocí je možné směrem dolů v hierarchii postupně ověřovat důvěryhodnost všech záznamů. Zůstává tak zachována původní hierarchická struktura DNS.

DNSSEC řeší i možnost podsouvání falešných informací o neexistenci domény pomocí záznamu typu NSEC [13] obsahujícího jméno podepsané domény, následující podle abecedy. Pokud je server dotazován na neexistující doménu, jako odpověď pošle informace o podepsané doméně, která by této neexistující doméně předcházela v abecedě. Pokud dotazovaná doména v abecedě leží mezi jménem poskytnutého záznamu a jménem v jeho NSEC záznamu, doména skutečně neexistuje. Např. uvažujme existenci domén *aa* a *ba*, při dotazu na neexistující doménu *ab* pošle server jako odpověď informace o doméně *aa*, jejíž NSEC záznam obsahuje *ba*. Z toho je patrné, že je informace o neexistenci domény věrohodná. Díky NSEC záznamům je však snadné vytvářet mapu zóny, což je zneužitelné pro cílení jiných typů útoků.

Tento problém byl odstraněn zavedením záznamu NSEC3 [14], který neobsahuje jméno následující domény. Obsahuje však haš jména toho záznamu, haš jména následujícího záznamu a parametry hašovací funkce. Jednotlivé haše jsou řazeny abecedně, princip ověřování je tak velmi podobný, pokud haš leží v intervalu mezi

dvěma haši v NSEC3, doména neexistuje. Jelikož spočítání původního jména z haše je výpočetně velmi náročné, je sestavení mapy zóny pomocí NSEC3 prakticky nemožné [15].

V praxi tuto validaci dopisů pomocí veřejných klíčů provádí tzv. validující resolver, který ověří autenticitu informací. Pokud podpis souhlasí, nastaví v hlavičce odpovědi takzvaný AD příznak (Authenticated Data). To značí, že zpráva nebyla podvržena [16].

1.3 Linuxové nástroje pro práci s DNS

Nejběžnějšími nástroji pro práci s DNS v Linuxu jsou *host* a *dig*. V obou případech se jedná o konzolové nástroje. Jejich pomocí je možné dotazovat se DNS serverů, použitím různých přepínačů lze dotazy blíže specifikovat. Například nastavit typ dotazu, verzi IP adresy kterou chceme získat, atd.

1.4 Linuxové konfigurační sooubory

Nejdůležitějším konfiguračním souborem je */etc/resolv.conf*, ve kterém jsou nastaveny IP adresy DNS serverů v textové podobě: *nameserver [IP adresa]*. V tomto souboru lze také přiřadit počítač do domény (*domain [doména]*).

Dalším souborem je soubor */etc/hosts*, v němž jsou uloženy statické DNS záznamy ve tvaru *[IP adresa] [doména]*.

2 Knihovny umožňující DNSSEC

Tato část práce pojednává o některých knihovnách umožňujících DNSSEC. Je v ní také zdůvodněna volba knihovny použité pro rozšíření Verse protokolu.

2.1 LIBEPP-NICBR

Jedná se o C++ knihovnu, která také částečně implementuje protokol EPP, používaný pro komunikaci mezi registrátory domén a doménovými registry [17]. Knihovna je vyvíjena na platformě FreeBSD 5.4 [18]. Knihovna však bez problémů funguje i na Linuxu [19] a Mac OS X [20].

Knihovna je kvalitně a celkem přehledně zdokumentována nástrojem doxygen [21] a její zdrojové kódy jsou také solidně okomentovány. Knihovna je poskytována pod BSD like licencí [22].

2.2 Libsresolv

Knihovna Libsresolv je vyvíjena francouzským institutem IRISA [23]. Je to rozšíření programu BIND, což je nejpoužívanější program implementující DNS server. Dokáže validovat DNS informace a zobrazovat cokoliv je obsaženo v DNS odpovědi. Knihovna je poskytována pod BSD like licencí. Vyžaduje knihovnu OpenSSL [24].

2.3 DNSruby

DNSruby je knihovna napsaná v interpretovaném skriptovacím jazyce Ruby [25], vyvíjená společností Nominet UK, která nabízí kompletní implementaci DNSSEC. Na oficiálních stránkách nabízí dokumentaci [26]. Knihovna je licencována Apache Licencí verze 2.0 [27].

2.4 DNSjava

DNSjava je knihovna implementující DNSSEC v jazyce Java [28], vyvíjená od roku 1999 Brianem Wellingtonem v rámci společnosti Nominum. Knihovna je poskytována

pod BSD licencí. Implementuje dotazy, přesměrovávání na jiné zóny a dynamické updatování. Může být využita v klientském programu a v minimální míře také pro server [29]. Implementuje také TSIG [30], jež je alternativou k DNSSEC [31]. Zdrojové kódy této knihovny jsou řádně okomentovány a je také k dispozici přehledná dokumentace vytvořená nástrojem Javadoc [32].

2.5 DNSpython

DNSpython je knihovna implementující DNSSEC v interpretovaném jazyce Python [33], je vyvíjena od roku 2003 Bobem Halleym, taktéž ze společnosti Nominum. Rozdíl oproti DNSjava je více méně pouze v jazyce, ve kterém je implementována. Co se týče dotazů, zón a updatování DNSpython poskytuje stejné možnosti jako DNSjava, zdrojové kódy jsou taktéž velmi dobře zdokumentovány [34]. Knihovna je dodávána pod BSD like licencí.

2.6 DNSSEC Toolkit

Jedná se o jednoduchý nástroj psaný v jazyce C, který umožňuje napsat jak programy pro resolver, tak také jednoduché nástroj pro ověření. V balíku ke stažení Toolkitu jsou i příklady implementace některých nástrojů. Vyvíjí jej Olivier Courtay z ENST Bretagne [35]. Je distribuována pod BSD licencí.

2.7 Net::DNS::SEC

Jedná se o rozšíření knihovny Net::DNS, vyvíjené Michaelem Fuhrem. Tato knihovna je psána v skriptovacím jazyce Perl [36]. Umožňuje použít téměř jakýkoliv DNS dotaz. Nezávisí na žádné knihovně v C, ale pro zvýšení rychlosti je možné jej slinkovat s knihovnou libsresolv. Vyvíjí ji Olaf Kolkman [37]. Knihovna je k dispozici pod upravenou BSD licencí.

2.8 Ldns

Knihovnu ldns vyvíjí od roku 2005 Nizozemská výzkumnická skupina NLnet Labs. Knihovna je psána v jazyce C. Umožňuje kompletně využívat DNS dotazy. Kompletně implementuje DNSSEC, jak ověřování, tak podepisování, kromě toho také implementuje TSIG. Podporuje IPv4 i IPv6 a obsahuje nástroj „Drill“, vytvořený pro ladění DNS dotazů [38].

Knihovna je vyvíjena především pro platformy Linux a FreeBSD, avšak úspěšně funguje i na platformách Mac OS X a Solaris [39]. Knihovna vyžaduje knihovnu Libtool [40], pro funkčnost DNSSEC knihovna vyžaduje knihovnu OpenSSL a pro funkčnost některých ukázkových aplikací knihovnu libpcap [42].

Tato knihovna byla vybrána k použití při realizaci zadání bakalářské práce, jelikož poskytuje veškerou potřebnou funkčnost, která je vyžadována, hlavně kompletní podporu DNSSEC. Díky tomu, že je napsána stejně jako Verse protocol v jazyce C, je rychlejší než knihovny psané v jiných jazycích. Další výhodou je její velmi dobrá dokumentace, jež díky dodatečným tutoriálům a vzorovým programům, předčí kvalitu jiných dobře dokumentovaných knihoven. Knihovna je umístěna ve standartních repositářích Linuxových distribucí a je tak snadno dosažitelná, licence umožňuje volné šíření a úpravy [41].

Některé knihovní funkce použité v knihovně Verse protokolu budou blíže popsány v kapitole 3.8.

3 Realizace

V této části je popsána vlastní realizace zadání a s ní související teorie, jsou v ní též ukázky zdrojových kódů.

Samotný Verse protokol je možno stáhnout na adrese *https://dev.nti.tul.cz/repos/verse2/*. DNSSEC verze protokolu se nachází na přiloženém CD a postup kompilace je popsán v příloze. Verse protokol používá k navázání spojení a autentizaci uživatele protokol TCP a k vlastnímu přenosu dat protokol UDP.

Jako vývojové prostředí bylo použito Eclipse [43]. Jedná se o Open source prostředí, které podporuje mnoho programovacích jazyků včetně C. Jako systém pro zprávu verzí bylo použito Subversion (SVN) [44]. SVN se skládá ze serverové části, kde jsou uloženy všechny verze zdrojových kódů a z klientské části, jež je integrována v Eclipse. Pomocí klientské části jsou nahrávány na server nové verze a stahovány aktualizace od jiných vývojářů.

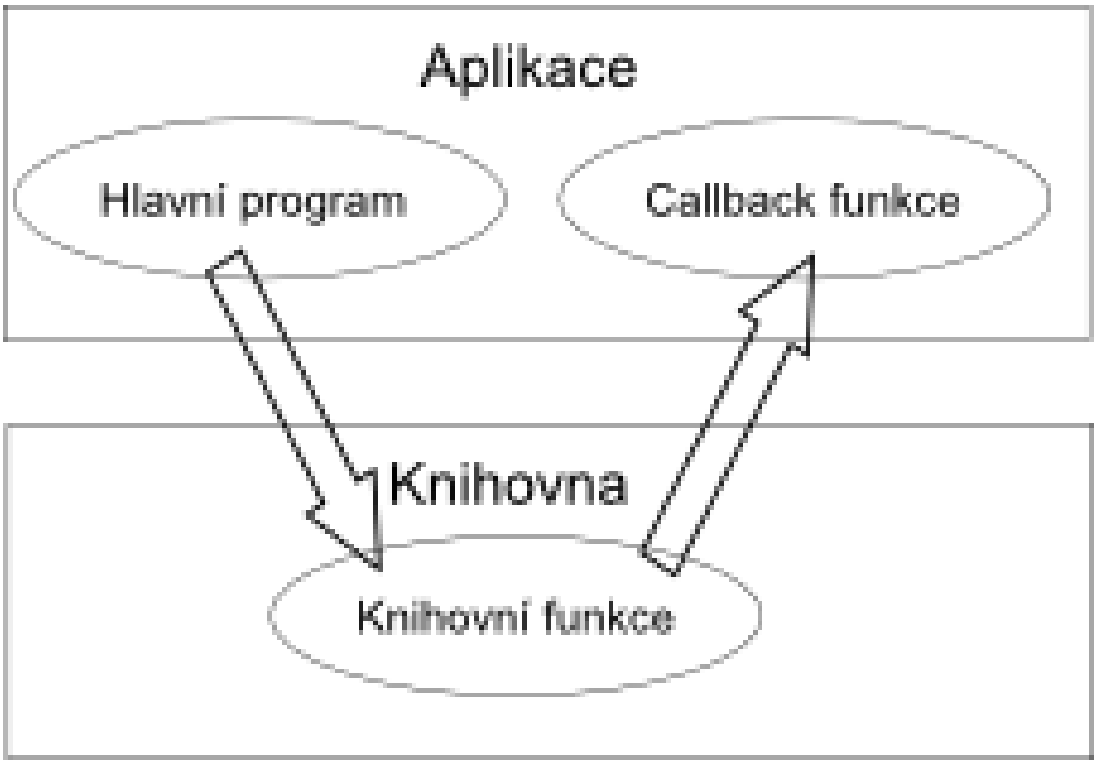
3.1 API

Application Programming Interface (API) je rozhraní pro programování aplikací. Jo to soubor funkcí a konstant nějaké knihovny, které při implementaci této knihovny může programátor používat v aplikaci.

3.2 Callback funkce

Callback funkce je používána v případě, kdy při běhu některé z knihovních funkcí, nastane událost, u které chceme, aby jí ošetřil programátor aplikace implementující tuto knihovnu. K tomu se využívají takzvané registrační funkce, které mají za úkol uložit ukazatel na funkci, která v aplikaci obsluhuje danou událost. Tak aby bylo možné v knihovní funkci volat funkci, definovanou v aplikaci implementující tuto knihovnu.

Registrační funkce má jediný vstupní parametr a tím je ukazatel na funkci s přesně danými vstupními parametry. Tato funkce je zahrnuta v API knihovny. V knihovně je struktura pro ukládání ukazatelů na jednotlivé callback funkce. Programátor aplikace vytvoří konkrétní implementaci funkce a tu pomocí registrační funkce zaregistruje



Obrázek 1: Volání callback funkce

do knihovny.

Když pak v knihovní funkci nastane událost, kterou chce programátor knihovny nechat ošetřit programátorem aplikace, zavolá se funkce, uložená na místě, kam ukazuje ukazatel získaný díky registrační funkci. Vykoná se callback funkce a poté se případně dále pokračuje ve vykonávání programu. Viz. Obrázek 1.

3.3 Certifikáty

Digitální certifikáty (dále TLS certifikáty) slouží k ověření důvěryhodnosti účastníka komunikace, se kterým je komunikováno prostřednictvím šifrovaného spojení. Verse protokol používá TLS [45] šifrování. Certifikáty fungují na principu digitálního podpisu, kdy veřejným klíčem vydaným certifikační autoritou, jsou podepisovány veřejné klíče účastníků šifrované komunikace. Funguje zde hierarchický přenos důvěry.

3.4 Umístění nových funkcí

DNS informaci bylo nutné ověřit ještě před navázáním TCP spojení. O navázání tohoto spojení se starají funkce v souboru *vc_tcp_connect.c*. Do něj byla umístěna funkce pro ověření DNS informace i funkce pro ověření TLS certifikátu. Funkce pro verifikaci DNS byla nazvána *vc_verify_dns* a její volání muselo proběhnout ještě před voláním funkce *vc_create_client_stream_conn*, která se stará o vlastní navázání TCP spojení a TLS handshake. Totu funkci bylo nutné upravit tak, aby bylo možné jí předat adresy, jež byly získány při předchozím DNS dotazování.

K tomu byla vytvořena v souboru *vc_tcp_connect.h* struktura *Addr_data*, jež je na následují ukázce zdrojového kódu. Jelikož v odpovědi od DNS serveru obvykle bývá více IP adres, struktura obsahuje IP adresu v textové formě a ukazatel na strukturu *Addr_data*, jež je odkazem na následující adresu.

```
typedef struct Addr_data {
    const char *addr;
    struct Addr_data *next;
}Addr_data;
```

3.5 Struktura pro ukládání callback funkcí

Jako první krok bylo nutné upravit již existující strukturu pro ukládání callback funkcí. Bylo potřeba ji rozšířit o další funkci, která slouží k předávání bezpečnostních dat. Jedním z hlavních požadavků na tuto funkci byla dostatečná univerzálnost, aby se dala použít nejenom k zobrazení DNS informací, ale byla schopna zobrazit například i informace o TLS certifikátu.

Ukládání callback funkcí je ve Verse protokolu řešeno v souborech: *v_func_storage.c* a v hlavičkovém souboru *v_func_storage.h*. V souboru *v_func_storage.c* je pouze inicializační funkce *initFuncStorage*. Do té bylo nutné přidat pouze řádek inicializující novou bezpečnostní funkci.

V hlavičkovém souboru *v_func_storage.h* je definována struktura pro ukládání callback funkcí. Ta se jmenuje *VFuncStorage* a jejími proměnnými jsou ukazatele na jednot-

livé callback funkce. Bylo ji nutné rozšířit o ukazatel na funkci s vhodnými parametry. Tato funkce byla nazvána *security_info*. Jejími parametry jsou ID relace, doménové jméno serveru, IP adresa serveru, textový výsledek a status výsledku. Následuje ukázka zdrojového kódu (rozšíření *VFuncStorage*).

```
void (*security_info)(const uint32 session_id,
        const char *peer_name,
        const char *peer_addr,
        const char *result,
        const uint8 status);
```

3.6 Registrační funkce

Jak již bylo v úvodu zmíněno, registrační funkce slouží k uložení callback funkce do struktury pro ukládání callback funkcí. V protokolu Verse jsou tyto funkce, pro klientskou část aplikace definovány v souboru *vc_connection.c*.

Nová registrační funkce byla nazvána *register_security_info* a jejím vstupním parametrem je ukazatel na funkci, jež má parametry jako callback funkce. V těle funkce ještě nad řádkem, ve kterém je ukazatel přiřazen k příslušné funkci ve struktuře *VFuncStorage*, je volání funkce *init_VC_CTX*. Tato funkce inicializuje strukturu *VC_CTX*, v níž se nachází mimo jiné právě struktura *VFuncStorage*. Inicializační funkce je ošetřena tak, že pokud je struktura již inicializována, znovu se inicializovat nebude. Následuje zdrojový kód registrační funkce.

```
void register_security_info(void (*func)(const uint32 session_id,
        const char *peer_name,
        const unsigned short ip_ver,
        const char *peer_addr,
        const char *result,
        const uint8 status))
{
    init_VC_CTX();
    vc_ctx->vfs.security_info = func;
}
```

3.7 Falešný příkaz

Falešný příkaz se liší od běžného příkazu tím, že není posílán po síti a slouží k volání callback funkcí, kdežto běžné příkazy jsou využívány pro komunikaci mezi klientem a serverem.

Falešné příkazy jsou zařazovány do fronty. Každý příkaz má svojí frontu, takže bylo nutné vytvořit nejen nový falešný příkaz, ale také novou frontu a strukturu k ukládání dat, pro tento příkaz.

Tyto příkazy a struktury pro ukládání dat k těmto příkazům, jsou definovány ve hlavičkovém souboru *v_command.h*. V tomto souboru bylo nutno definovat konstantu pro nový falešný příkaz. Nazvána byla *FAKE_CMD_SECURITY_INFO* a byla jí přiřazena hodnota 3. Struktura pro ukládání bezpečnostních dat byla nazvána *Security_Data*. Její atributy v podstatě kopírují parametry callback funkce, pouze tu chybí číslo relace a kvůli kompatibilitě s ostatními Verse příkazy, je tu navíc ID příkazu. Následuje ukázka zdrojového kódu struktury pro ukládání bezpečnostních dat.

```
/* Structure for storing DNS or TLS data in incoming queue */
typedef struct Security_Data {
    /* Fake cmd_id for compatibility with node commands */
    uint8                       id;
    char                        *peer_name;
    unsigned short              ip_ver;
    char                        *peer_addr;
    char                        *result;
    uint8                       status;
} SecurityData;
```

Fronty příkazů jsou definovány v souboru *v_dataqueue.c*. O vytvoření front se stará inicializační funkce *v_queue_init*. V ní bylo opět nutné přidat příkaz *FAKE_CMD_SECURITY_INFO*.

Velikost prvku (item_size) je nastavena na velikost struktury *Security_Data*, parametry adresy (addr_offset a addr_size) jsou nastaveny na nulu, jelikož se jedná o falešný, tedy offline příkaz. Hašovací funkce (hash_function) je nastavena na *blind_hash_func*, jelikož předpokládáme, že callback funkce bude během připojování volána dvakrát s různými parametry. Tato funkce umožňuje vložení více příkazů do fronty za sebe.

Načítání příkazů z callback funkcí je prováděno zavoláním funkce *verse_callback_update* umístěné v souboru *vc_connection.c*. Tato funkce byla rozšířena o načtení fronty bezpečnostních příkazů (ukázka kódu v příloze A).

Funkce má jediný vstupní parametr, jímž je ID relace. Funkce prochází všemi relacemi. Jakmile narazí na požadovanou relaci, začne postupně z front příkazů načítat jednotlivé příkazy. První na řadě je fronta, která byla přidána v rámci této práce, tedy fronta pro bezpečnostní příkazy. Jsou-li ve frontě příkazy, jsou postupně načteny, a je volána příslušná callback funkce, uložená ve *VFuncStorage*. Následuje ukázka kódu inicializace fronty falešných příkazů.

```
case FAKE_CMD_SECURITY_INFO:    /* Fake command for showing DNS or TLS info */
    queue->item_size = sizeof(struct Security_Data);
    queue->addr_offset = 0;
    queue->addr_size = 0;
    queue->hash_function = blind_hash_func;
    queue->hash_table = NULL;
    break;
```

3.8 Ověření DNS záznamu

Jak již bylo zmíněno v úvodu kapitoly, o ověření DNS záznamu se stará funkce *vc_verify_dns*, umístěná v souboru *vc_tcp_connect.c*. Funkce má jako vstupní parametr ukazatel na strukturu *VSession*, jež obsahuje informace o aktuální relaci, a výstupní parametr ukazatel na strukturu *Security_Data*, tedy strukturu pro falešný příkaz. Funkce vrací strukturu *Addr_data*, v níž jsou uloženy IP adresy, získané při DNS dotazech. Viz. následující ukázka zdrojového kódu.

```
struct Addr_data vc_verify_dns(struct VSession *session,
        struct Security_Data *security_data)
```

Při vykonávání funkce jsou nejprve získány a ověřeny adresy IPv4, ukázka zdrojového kódu v příloze A. Poté IPv6 a dále je falešný příkaz s patřičnými daty přidán do fronty příkazů a jsou vráceny IP adresy, získané dotazy na DNS server.

V této funkci je využíváno funkcí knihovny Ldns, jako první *ldns_resolver_new_frm_file*. Pomocí této funkce se nastavuje, jakého DNS serveru se chceme dotazovat.Výstupním parametrem je struktura *ldns_resolver*, v níž je konfigurace DNS. Vstupní parametr je soubor, z něhož mají být informace načteny. Pokud je zadáno *NULL* je použit soubor */etc/grub.conf*. Pokud vše proběhne v pořádku funkce vrací *LDNS_STATUS_OK*. V opačném případě funkce vrátí chybový kód.

Další použitou funkcí je *ldns_dname_new_frm_str*, vracející strukturu *ldns_rdf* a se vstupním parametrem ukazatelem na znak (char). Tato funkce vytváří z textového řetězce informace o doméně, které jsou používány jako parametr DNS dotazů.

Funkce *ldns_resolver_set_dnssec* má vstupní parametry *ldns_resolver* a *bool*. Její pomocí se nastavuje zda má být zapnuta podpora DNSSEC.

Další funkcí je *ldns_resolver_query*, posílající dotazy na DNS server. Vrací strukturu *ldns_pkt* reprezentující DNS zprávu a má pět vstupních parametrů, struktury *ldns_resolver*, a *ldns_rdf*, druh záznamu, třídu dotazu a příznak. Pomocí těchto parametrů je možné přesně specifikovat dotaz, tedy na jaký server má být dotaz poslán a jaký druh záznamu je požadován (např. adresa IPv4). Ve Verse protokolu byl u dotazu použit příznak RD, tedy po DNS serveru je požadováno rekurzivní doptávání.

Věrohodnost dat je kontrolována funkcí *ldns_pkt_ad*, jež kontroluje, zda odpověď obsahuje příznak AD. Ten značí, že jsou data obsažená v paketu autentická. Vstupním parametrem je struktura *ldns_pkt*. Funkce vrací *true*, pokud paket příznak obsahuje. V opačném případě vrací *false*.

Z odpovědi bylo potřeba vybrat požadované RR záznamy, k čemuž slouží funkce *ldns_pkt_rr_list_by_type* vracející strukturu *ldns_rr_list*. Vstupními parametry jsou *ldns_pkt*, typ záznamu a sekce paketu, z níž mají být záznamy vybrány. Poslední použitou funkcí je *ldns_rdf2str*, která převádí IP adresu z binárního formátu do textového.

3.9 Ověření TLS certifikátu

Podobně jako u DNS byla i pro ověření TLS certifikátu vytvořena nová funkce v souboru *vc_tcp_connect.c*. Tato funkce byla nazvána *vc_verify_cert*. Jako vstupní pa-

rametr má ukazatel na strukturu *vContext*, jež obsahuje mimo jiné strukturu *VSession*, použitou pro předchozí funkci. Ale hlavně obsahuje strukturu *VStreamConn*, která obsahuje informace o TLS spojení a je z ní tedy možno získat TLS certifikát serveru. Výstupním parametrem je stejně jako u předchozí funkce ukazatel na strukturu *Security_Data*, na následují ukázce zdrojového kódu je hlavička funkce *vc_verify_cert*.

```
void vc_verify_cert(struct vContext *C, struct Security_Data *security_data)
```

K ověření TLS certifikátu jsou použity funkce knihovny OpenSSL. První z nich je funkce *SSL_get_peer_certificate* použitá k získaní certifikátu serveru. Jejím vstupním parametrem je struktura *SSL* a vrací strukturu *X509*, která představuje TLS certifikát. Z této struktury, jsou pomocí funkcí *X509_get_subject_name* a *X509_NAME_oneline* získány informace o certifikátu v textová podobě, ty jsou pak předány bezpečnostní callback funkci.

Nejdůležitější funkcí ve *vc_verify_cert* je *SSL_get_verify_result*. Tato funkce se stará o vlastní ověření certifikátu. Jako vstupní parametr má strukturu *SSL* a vrací chybové kódy, ty jsou předány bezpečnostní callback funkci. Podle chybového kódu je doplněna textová informace upřesňující chybu a ta je také předána bezpečnostní callback funkci. Pokud je certifikát v pořádku, funkce vrátí 0. Bezpečnostní callback funkci je pak předán kód *SECURITY_CERT_OK*.

3.10 Rozšíření Verse API

Verse API je definováno v souboru *verse.h*. Bylo jej nutné rozšířit především o registrační funkci *register_security_info*. Tak aby programátor používající knihovnu Verse protokolu mohl zaregistrovat příslušnou callback funkci. Dále bylo Verse API rozšířeno o konstanty, jež jsou používány jako chybové kódy pro bezpečnostní funkci, viz následující ukázka zdrojového kódu.

```
/* Security status */
#define SECURITY_DNS_OK     0
#define SECURITY_CERT_OK    1
#define SECURITY_CERT_NON   200
#define SECURITY_DNS_FAULT  201
```

3.11 Vzorová aplikace

Jako vzorová aplikace byla použita konzolová aplikace „Example of Verse client“, čili příklad Verse klienta. Ta byla již napsána Ing. Jiřím Hnídkem. Bylo samozřejmě nutné ji rozšířit o konkrétní implementaci callback funkce, viz. následují ukázka zdrojového kódu.

```
void cb_security_info(const uint32 session_id, const char *peer_name,
        const unsigned short ip_ver, const char *peer_addr, const char *result,
        const uint8 status)
{
    char con;
    printf("Session info:\n");
    printf("Session: %d\tName: %s\tAddress: %s\n", session_id, peer_name,
            peer_addr);
    printf("%s\n", result);
    if ((status != SECURITY_CERT_OK) && (status != SECURITY_DNS_OK)) {
        printf("Continue? (y/n)\n");
        scanf("%c", &con);
        if (con != 'y') {
            exit(EXIT_SUCCESS);
        }
        getchar();
    }
}
```

Vstupními parametry funkce jsou pochopitelně ID relace, jméno serveru, IP adresa, potažmo adresy, v textové formě, textová informace a výsledný status ověřování. Funkce během svého vykonávání vypíše do konzole informace o relaci a informace o DNS, potažmo o certifikátu. Pokud status neobsahuje *SECURITY_DNS_OK* nebo *SECURITY_CERT_OK*, je uživatel dotázán, zda má program pokračovat v připojování se k serveru, V opačném případě program pokračuje automaticky.

Tato funkce je v průběhu vykonávání aplikace volána dvakrát. Poprvé zobrazuje informaci o validitě DNS záznamu, podruhé informace o TLS certifikátu.

Bylo také nutné zaregistrovat tuto funkci, pomocí registrační funkce, jež byla přidána do Verse API. To je provedeno v následující ukázce zdrojového kódu. Tím bylo dosaženo funkcionality vzorové aplikace, potřebné k demonstrování funkčnosti změn, provedených ve Verse protokolu.

```
register_security_info(cb_security_info);
```

4 Testování

Jelikož nebyla možnost nainstalovat Verse server na serveru, který má podepsanou doménu, testování funkčnosti na vzorové aplikaci, konkrétně ověření DNS záznamu, proběhlo bez vlastního připojení k Verse serveru. Avšak k vlastnímu otestování funkčnosti verifikace DNS záznamu není připojení k Verse serveru nutné. Stačí pokusit se připojit pomocí klientského programu k jiným serverům. DNS záznam je totiž ověřován ještě před vlastním navázáním komunikace ze serverem.

V mém případě proběhlo testování připojením k podepsaným serverům *www.nic.cz* a *www.dnssec.cz* a nepodepsanému serveru *www.tul.cz*. Test připojení k Verse serveru byl proveden lokálně a to pro IPv4 (localhost) i pro IPv6 (localhost6). Rovněž bylo otestováno zadání neexistující adresy.

4.1 Podepsané domény

První test proběhl připojením vzorové aplikace k serveru *www.nic.cz*, viz. následující ukázka.

```
[ZDN@PanDELL Bakalarka]$ ./bin/verse_client www.nic.cz
Session info:
Session: 0      Name: www.nic.cz        Address: 217.31.205.50, 2002:d91f
:cd32::1, 2001:1488:0:3::2,
Authentic address
```

Na screenshotu jsou vidět zobrazené informace o relaci: její číslo, jméno serveru a IP adresy. V tomto případě jedna IPv4 a dvě IPv6. Dále je vypsána informace o tom, že DNS záznamy jsou validní. Díky tomu aplikace pokračuje pokusem připojit se k portu 12345 (na něm naslouchá Verse server), serveru *www.nic.cz*. Spojení samozřejmě nelze navázat, jelikož na serveru *www.nic.cz* Verse server nainstalován a spuštěn není. Obdobný výsledek vznikne i při pokusu připojit se k serveru *www.dnssec.cz* (následují ukázka). S tím rozdílem, že zde byla získána pouze jedna adresa IP verze 6.

```
[ZDN@PanDELL Bakalarka]$ ./bin/verse_client www.dnssec.cz
Session info:
Session: 0      Name: www.dnssec.cz     Address: 217.31.205.51, 2001:1488
:0:3::5,
Authentic address
```

4.2 Neexistující doména

Ve třetím případě proběhlo zadání neexistující domény (na ukázce níže), tak aby bylo demonstrováno, že knihovna Verse protokolu i po přidání nové funkce po této události, zůstane stabilní.

```
[ZDN@PanDELL Bakalarka]$ ./bin/verse_client nesmysl
Session info:
Session: 0      Name: nesmysl   Address:
Host not found
Continue? (y/n)
```

Na obrázku jsou samozřejmě oproti předchozím případům změny. Není zobrazena žádná IP adresa a je vypsána informace o tom že server nebyl nalezen. Aplikace se také dotazuje, zda se má pokusit pokračovat v navazování spojení.

4.3 Nepodepsaná doména

Další test proběhl obdobně, jako předchozí, tedy pokusem o připojení k serveru, tentokrát k doméně, která sice existuje, ale není podepsána. Zvolena byla doména *www.tul.cz*, výsledek je na další ukázce.

```
[ZDN@PanDELL Bakalarka]$ ./bin/verse_client www.tul.cz
Session info:
Session: 0      Name: www.tul.cz        Address: 147.230.16.27, 2001:718:
1c01:16:216:3eff:fe1a:d23f,
Non authentic address
Continue? (y/n)
```

Výsledek je podobný jako u prvních dvou testů. Jsou opět vypsány informace o relaci, však tentokrát je DNS záznam neautentický. Aplikace se tudíž dotáže, zda má pokračovat v připojování se k serveru. Po kladné odpovědi se připojení k portu 12345 opět nepodaří, Verse server zde není nainstalován.

4.4 Lokální test připojení

Další testy už probíhaly lokálně. Bylo potřeba spustit Verse server, jež naslouchá na portu 12345, nejprve připojením skrze IP verze 4, a poté skrze IP verze 6. V obou případech je hlášena neautentická adresa, jelikož *localhost* ani *localhost6* nejsou podepsány. Po volání funkce pro ověření certifikátu, je hlášena další chyba a to self signed certifikát, tedy certifikát podepsaný sám sebou. Je opět zvoleno pokračovat. Po zadání uživatelského jména a hesla se aplikace úspěšně přihlásí k Verse serveru. Následuje ukázka spuštění Verse serveru, poté připojení přes IPv4 a IPv6.

```
[ZDN@PanDELL Bakalarka]$ ./bin/verse_server
WARNING: was not able to read file: /etc/verse
WARNING: ... using default values
DEBUG: Added: username: rachel, ID: 1000, realname: Rachel Green
DEBUG: Added: username: monica, ID: 1001, realname: Monica Geller
DEBUG: Added: username: phoebe, ID: 1002, realname: Phoebe Buffay
DEBUG: Added: username: joey, ID: 1003, realname: Joey Tribbiani
DEBUG: Added: username: chandler, ID: 1004, realname: Chandler Bing
DEBUG: Added: username: ross, ID: 1005, realname: Ross Geller
DEBUG: Added: username: testing, ID: 1006, realname: n/a
DEBUG: 7 user account loaded from file: ./config/users.csv
DEBUG:  +0s     Server listen on TCP port: 12345
DEBUG: Wait for data ...
DEBUG:  +1s     Server listen on TCP port: 12345
DEBUG:  +2s     Server listen on TCP port: 12345
```

```
[ZDN@PanDELL Bakalarka]$ ./bin/verse_client localhost
Session info:
Session: 0      Name: localhost Address: 127.0.0.1,
Non authentic address
Continue? (y/n)
y
DEBUG: Try to connect to: localhost:12345
DEBUG: SSL connection established.
DEBUG: SSL connection uses: AES256-SHA cipher.
DEBUG: Client TCP state: USRAUTH_none
Session info:
Session: 0      Name: localhost Address: 127.0.0.1
Certificate info:
/C=AU/ST=Some-State/O=Internet Widgits Pty Ltd
Self signed certificate
The passed certificate is self signed and the same certificate cannot be
found in the list of trusted certificates.
Continue? (y/n)
y
cb_receive_user_authenticate() username: , auth_methods_count: 0, methods
:
Username: testing
DEBUG: Send message: Socket: 3, 127.0.0.1:12345 Ver: 1, Len: 15
        USER_AUTH_REQUEST, Username: testing, Type: None
DEBUG: Receive message: Socket: 3, 127.0.0.1:12345 Ver: 1, Len: 7
        USER_AUTH_FAILURE, Methods: Password,
DEBUG: Client TCP state: USRAUTH_data
cb_receive_user_authenticate() username: testing, auth_methods_count: 1,
methods: 2,
Password:
DEBUG: Send message: Socket: 3, 127.0.0.1:12345 Ver: 1, Len: 21
        USER_AUTH_REQUEST, Username: testing, Type: Password: *****
DEBUG: Receive message: Socket: 3, 127.0.0.1:12345 Ver: 1, Len: 66
        USER_AUTH_SUCCESS, UserID: 1006, AvatarID: 65537
```

```
[ZDN@PanDELL Bakalarka]$ ./bin/verse_client localhost6
Session info:
Session: 0      Name: localhost6        Address: ::1,
Non authentic address
Continue? (y/n)
y
DEBUG: Try to connect to: localhost6:12345
DEBUG: SSL connection established.
DEBUG: SSL connection uses: AES256-SHA cipher.
DEBUG: Client TCP state: USRAUTH_none
Session info:
Session: 0      Name: localhost6        Address: ::1
Certificate info:
/C=AU/ST=Some-State/O=Internet Widgits Pty Ltd
Self signed certificate
The passed certificate is self signed and the same certificate cannot be
found in the list of trusted certificates.
Continue? (y/n)
y
cb_receive_user_authenticate() username: , auth_methods_count: 0, methods
:
Username: testing
DEBUG: Send message: Socket: 3, [::1]:12345 Ver: 1, Len: 15
        USER_AUTH_REQUEST, Username: testing, Type: None
DEBUG: Receive message: Socket: 3, [::1]:12345 Ver: 1, Len: 7
        USER_AUTH_FAILURE, Methods: Password,
DEBUG: Client TCP state: USRAUTH_data
cb_receive_user_authenticate() username: testing, auth_methods_count: 1,
methods: 2,
Password:
DEBUG: Send message: Socket: 3, [::1]:12345 Ver: 1, Len: 21
        USER_AUTH_REQUEST, Username: testing, Type: Password: *****
DEBUG: Receive message: Socket: 3, [::1]:12345 Ver: 1, Len: 66
        USER_AUTH_SUCCESS, UserID: 1006, AvatarID: 65539
```

4.5 Nedostatky, možná budoucí vylepšení

Tato část práce pojednává o možných budoucích vylepšeních a o nedostatcích současné implementace. Ideální není že je v knihovně protokolu Verse po ověření DNS informací volána funkce *verse_callback_update*. Ta by měla být volána pouze vně knihovny, ovšem z důvodu včasného předání informací o DNS skrze callback funkci je volána uvnitř knihovny, rozhodně tedy stojí za úvahu, jak to v budoucnu řešit.

Taktéž stojí za úvahu zda, případně jakým způsobem v budoucnu nedodefinovat, či nepředefinovat konstanty používané v bezpečnostní callback funkci. V té jsou nyní kromě konstant definovaných ve *verse.h*, používány i chybové kódy knihovny OpenSSl (tato knihovna je v současné době nutná ke kompilaci knihony Verse protokolu).

5 Závěr

Tato bakalářská práce popisuje rozšíření Verse protokolu o DNSSEC. Součástí práce je stručná rešerše o knihovnách umožňujících DNSSEC, pro samotnou implementaci pak byla vybrána knihovna ldns.

Do knihovny Verse protokolu byly implementovány funkce ověřující autenticitu DNS záznamu a TLS certifikát. Dále byl přidán bezpečnostní falešný příkaz a callback funkce. Byly přidány, nebo upraveny struktury, jako struktura pro ukládání callback funkcí, IP adres a struktura falešného příkazu. Také byly upraveny některé již existující funkce.

Cílem práce bylo také vhodné rozšíření Verse API. To bylo rozšířeno o některé konstanty a registrační funkci, registrující bezpečnostní callback funkci. Díky tomu je možné registrovat callback funkci a používat příslušné konstanty v aplikacích implementujících Verse protokol.

Jako vzorová aplikace byla využita již existující klientská aplikace. Ta byla rozšířena o bezpečnostní callback funkci. Na rozšířené vzorové aplikaci pak probíhalo testování funkčnosti implementace DNSSEC.

Testování vzorové aplikace bylo úspěšné, adresy podepsaných domén byli správně vyhodnoceny jako autentické, naopak adresa nepodepsané domény byla vyhodnocena jako neautentická. Test připojení k lokálně spuštěnému Verse serveru byl také úspěšný.

Reference

[1] *HNÍDEK, Jiří.* **Síťový protokol pro grafické aplikace.** Liberec, 2011. 171 s. Dizertační práce. Technická univerzita v Liberci.

[2] Ietf.org [online]. 1999 [cit. 2011-05-18]. **Domain Name System Security Extensions RFC 2535.** Dostupné z WWW: http://datatracker.ietf.org/doc/rfc2535/.

[3] Ietf.org [online]. 1987 [cit. 2011-05-18]. **Domain names - implementation and specification RFC 1035.** Dostupné z WWW: http://datatracker.ietf.org/doc/rfc1035/.

[4] Ietf.org [online]. 1981 [cit. 2011-05-18]. **Internet Protocol RFC 791.** Dostupné z WWW: http://datatracker.ietf.org/doc/rfc791/.

[5] Nic.cz [online]. 2011 [cit. 2011-05-18]. **O DNSSEC.** Dostupné z WWW: http://www.nic.cz/dnssec/.

[6] **Domain Name System.** In Wikipedia : the free encyclopedia [online]. St. Petersburg (Florida) : Wikipedia Foundation, [cit. 2011-05-18]. Dostupné z WWW: http://cs.wikipedia.org/wiki/Domain_Name_System.

[7] *SURÝ, Ondřej.* **DNSSEC a DNS zpráva pod drobnohledem.** DNSSEC a bezpečné DNS [online]. 2008, 4, [cit. 2011-05-18]. Dostupný z WWW: http://www.root.cz/clanky/dnssec-a-dns-zprava-pod-drobnohledem/.

[8] Ietf.org [online]. 1980 [cit. 2011-05-18]. **User Datagram Protocol RFC 768.** Dostupné z WWW: http://datatracker.ietf.org/doc/rfc768/.

[9] Ietf.org [online]. 1981 [cit. 2011-05-18]. **Transmission Control Protocol RFC 793.** Dostupné z WWW: http://datatracker.ietf.org/doc/rfc793/.

[10] *ŠŤASTNÝ, Petr.* Dns-info.cz [online]. 2007 [cit. 2011-05-18]. **Komunikace přes UDP a TCP - DNS.**
Dostupné z WWW: http://www.dns-info.cz/dns/protokol-tcp-udp.html.

[11] Ietf.org [online]. 2001 [cit. 2011-05-18]. **Electronic Signature Policies RFC 3125.**
Dostupné z WWW: http://datatracker.ietf.org/doc/rfc3125/.

[12] *SATRAPA, Pavel.* **Věrohodné DNS čili DNSSEC.**
Lupa.cz [online]. 26. 5. 2005, [cit. 2011-05-11]. Dostupný z WWW: http://www.lupa.cz/clanky/verohodne-dns-cili-dnssec/

[13] Ietf.org [online]. 2005 [cit. 2011-05-18]. **Resource Records for the DNS Security Extensions RFC 4034.**
Dostupné z WWW: http://datatracker.ietf.org/doc/rfc4034/.

[14] Ietf.org [online]. 2008 [cit. 2011-05-18]. **DNS Security (DNSSEC) Hashed Authenticated Denial of Existence RFC 5155.**
Dostupné z WWW: http://datatracker.ietf.org/doc/rfc5155/.

[15] *SATRAPA, Pavel.* Lupa.cz [online]. 2008 [cit. 2011-05-18]. **NSEC3 – DNSSEC, který nic nevyzradí.** Dostupné z WWW:
http://www.lupa.cz/clanky/nsec3nbspndash-dnssec-ktery-nic-nevyzradi/.

[16] *SURÝ, Ondřej.* **Jak funguje DNSSEC?**
Seriál DNSSEC a bezpečné DNS [online]. 29. 12. 2008, č. 4, [cit. 2011-05-11].
Dostupný z WWW: http://www.root.cz/serialy/dnssec-a-bezpecne-dns/

[17] Registro.br [online]. 2010 [cit. 2011-05-11]. **Libepp-nicbr.**
Dostupné z WWW: http://registro.br/epp/index-EN.html

[18] Www.cz.freebsd.org [online]. 2011 [cit. 2011-05-19]. **Projekt FreeBSD.**
Dostupné z WWW: http://www.cz.freebsd.org/cs/.

[19] Www.linux.cz [online]. 2011 [cit. 2011-05-19].
České stránky systému GNU/Linux. Dostupné z WWW: http://www.linux.cz/.

[20] Apple.com [online]. 2011 [cit. 2011-05-19]. **Apple - Mac OS X.**
Dostupné z WWW: http://www.apple.com/cz/macosx/.

[21] Registro.br [online]. 2010 [cit. 2011-05-11]. **Libepp-nicbr.**
Dostupné z WWW: http://registro.br/epp/docs-EN.html

[22] Linfo.org [online]. 2004 [cit. 2011-05-19]. **BSD License Definition.**
Dostupné z WWW: http://www.linfo.org/bsdlicense.html.

[23] Idsa.irisa.fr [online]. 2004 [cit. 2011-05-11]. **Projet IDsA.**
Dostupné z WWW: http://idsa.irisa.fr/index.php?page=libsresolv%5C&lang=en

[24] Openssl.org [online]. 2011 [cit. 2011-05-19]. **OpenSSL.**
Dostupné z WWW: http://www.openssl.org/.

[25] Ruby-lang.org [online]. 2011 [cit. 2011-05-19]. **Ruby Programming Language.**
Dostupné z WWW: http://www.ruby-lang.org/en/.

[26] Dnsruby.rubyforge.org [online]. 2011 [cit. 2011-05-11]. **RDoc Documentation.**
Dostupné z WWW: http://dnsruby.rubyforge.org/

[27] Apache.org [online]. 2004 [cit. 2011-05-19]. **Apache License, Version 2.0.**
Dostupné z WWW: http://www.apache.org/licenses/LICENSE-2.0.

[28] Www.java.com [online]. 2011 [cit. 2011-05-19]. **Java.**
Dostupné z WWW: http://www.java.com/en/.

[29] Dnsjava.org [online]. 2011 [cit. 2011-05-11]. **Dnsjava.**
Dostupné z WWW: http://www.dnsjava.org/

[30] Ietf.org [online]. 2000 [cit. 2011-05-19]. **Secret Key Transaction Authentication for DNS (TSIG) RFC 2845.**
Dostupné z WWW: http://datatracker.ietf.org/doc/rfc2845/.

[31] **Tsig.** In Wikipedia : the free encyclopedia [online]. St. Petersburg (Florida) : Wikipedia Foundation, [cit. 2011-05-11].
Dostupné z WWW: http://cs.wikipedia.org/wiki/Tsig

[32] Dnsjava.org [online]. 2011 [cit. 2011-05-11]. **Dnsjava**.
Dostupné z WWW: http://www.dnsjava.org/dnsjava-current/doc/

[33] Python.org [online]. 2011 [cit. 2011-05-19]. **Python Programming Language.**
Dostupné z WWW: http://www.python.org/.

[34] www.dnspython.org [online]. 2011 [cit. 2011-05-11]. **Dnspython home page.**
Dostupné z WWW: http://www.dnspython.org/

[35] Dnssec.net [online]. 2011 [cit. 2011-05-11]. **DNSSEC Software, DNSSEC Tools, DNSSEC Utilities.**
Dostupné z WWW: http://www.dnssec.net/software

[36] Perl.org [online]. 2011 [cit. 2011-05-19]. **The Perl Programming Language.**
Dostupné z WWW: http://www.perl.org/.

[37] Www.net-dns.org [online]. 2011 [cit. 2011-05-11]. **NET::DNS.**
Dostupné z WWW: http://www.net-dns.org/

[38] Nlnetlabs.nl [online]. 2011 [cit. 2011-05-11]. **Ldns.**
Dostupné z WWW: http://www.nlnetlabs.nl/projects/ldns/

[39] Oracle.com [online]. 2011 [cit. 2011-05-19]. **Oracle Solaris.** Dostupné z WWW: http://www.oracle.com/cz/products/servers-storage/solaris/index.html.

[40] Gnu.org [online]. 2010 [cit. 2011-05-19]. **GNU Libtool.**
Dostupné z WWW: http://www.gnu.org/software/libtool/.

[41] Nlnetlabs.nl [online]. 2006 [cit. 2011-05-19]. **License.**
Dostupné z WWW: http://nlnetlabs.nl/svn/ldns/trunk/LICENSE.

[42] Fedoraproject.org [online]. 2011 [cit. 2011-05-19]. **Libpcap.**
Dostupné z WWW: https://admin.fedoraproject.org/pkgdb/acls/name/libpcap.

[43] Eclipse.org [online]. 2011 [cit. 2011-05-19]. **Eclipse.**
Dostupné z WWW: http://www.eclipse.org/.

[44] Apache.org [online]. 2011 [cit. 2011-05-19]. **Subversion.**
Dostupné z WWW: http://subversion.apache.org/.

[45] Ietf.org [online]. 1999 [cit. 2011-05-19]. **The TLS Protocol Version 1.0 RFC 2246.**
Dostupné z WWW: http://datatracker.ietf.org/doc/rfc2246/.

Příloha A - Ukázky zdrojových kódů

Zkrácená funkce verse_callback_update

```
/* Go through all sessions ... */
for (i = 0; i < vc_ctx->max_sessions; i++) {
    /* ... and try to find connection with session_id */
    if (vc_ctx->vsessions[i] != NULL && vc_ctx->vsessions[i]->session_id
            == session_id) {
        /* Pop all data of incoming messages from queues */

        /* Security Queue */
        while (v_queue_pop_item(
                vc_ctx->vsessions[i]->in_queue->queues[FAKE_CMD_SECURITY_INFO],
                &security_data) != NULL) {
            /* Call callback function */
            vc_ctx->vfs.security_info(session_id, security_data.peer_name,
                    security_data.ip_ver, security_data.peer_addr,
                    security_data.result, security_data.status);
        }

        break;
    }
}
```

Získání a ověření DNS záznamu IPv4

```
/* verify IPv4 address */
if (strcasecmp(session->peer_hostname, "localhost6") != 0) {
    packet = ldns_resolver_query(resolver, domain, LDNS_RR_TYPE_A,
            LDNS_RR_CLASS_IN, LDNS_RD);
    if (packet) {
        if (ldns_pkt_ad(packet)) {
            security_data->status = SECURITY_DNS_OK;
            security_data->result = "Authentic address";
        } else {
            security_data->status = SECURITY_DNS_FAULT;
            security_data->result = "Non authentic address";
        }
        /* get RRs */
        rr_list = ldns_pkt_rr_list_by_type(packet, LDNS_RR_TYPE_A,
                LDNS_SECTION_ANSWER);
        if (rr_list) {
            /* go trough RRs */
            for (i = 0; i < rr_list->_rr_count; i++) {
                rr = rr_list->_rrs[i];
                ip = *rr->_rdata_fields;
                ldns = ldns_rdf2str(ip);
                strcat(addrs, ldns);
                strcat(addrs, ", ");
                next = verified_addrs;
                verified_addrs = (struct Addr_data *) malloc(
                        sizeof(struct Addr_data));
                verified_addrs->addr = ldns;
                verified_addrs->next = next;
            }
            security_data->ip_ver = AF_INET;
        }
    }
}
```

Změny IPv6 oproti IPv4

```
packet = ldns_resolver_query(resolver, domain, LDNS_RR_TYPE_AAAA,
        LDNS_RR_CLASS_IN, LDNS_RD);


  /* get RRs */
  rr_list = ldns_pkt_rr_list_by_type(packet, LDNS_RR_TYPE_AAAA,
          LDNS_SECTION_ANSWER);


       if (ldns_pkt_ad(packet) && (security_data->status
               == SECURITY_DNS_OK)) {
           security_data->status = SECURITY_DNS_OK;
           security_data->result = "Authentic address";
       } else {
           security_data->status = SECURITY_DNS_FAULT;
           security_data->result = "Non authentic address";
       }
```

Ověření TLS certifikátu

```
/* get and verify certificate */
cert = SSL_get_peer_certificate(C->stream_conn->io_ctx.ssl);
if (cert != NULL) {
    line = X509_NAME_oneline(X509_get_subject_name(cert), 0, 0);
    strcat(result, line);
    if ((status = SSL_get_verify_result(C->stream_conn->io_ctx.ssl)) > 1) {
        security_data->status = (uint8) status;
        switch (status) {
        case X509_V_ERR_UNABLE_TO_GET_ISSUER_CERT:
            strcat(result,
                    "\nUnable to get issuer certificate.\nThe issuer"
                        " certificate of a looked up certificate could not"
                        " be found. This normally means the list of "
                        "trusted certificates is not complete.");
            break;
        case X509_V_ERR_UNABLE_TO_GET_CRL:
            strcat(result, "\nUnable to get certificate CRL\n"
                "The CRL of a certificate could not be found.");
            break;
        default:
            strcat(result, "\nCertificate error");
            break;
        }
    } else {
        /* certificate ok */
        security_data->status = SECURITY_CERT_OK;
        strcat(result, "\nCertificate OK");
    }
} else {
    /* no certificate provided */
    security_data->status = SECURITY_CERT_NON;
    strcat(result, "\nNo certificate");
}
```

Příloha B - Informace k přiloženému CD

K bakalářské práci je přiloženo CD se zdrojovými kódy Verse protokolu. Pro úspěšnou instalaci je nutné mít nainstalován modul PAM a knihovny OpenSSL a ldns. Knihovnu ldns je možno stáhnout na adrese *www.nlnetlabs.nl/projects/ldns/*.

Verse klient ani server není nutné instalovat, stačí je pouze zkopírovat na disk, případně sestavit použitím příkazu *make*. Před spuštěním serveru je nutné nastavit seznam uživatelů a uložit jej do adresáře *./config*, jako *users.csv*. V tomto adresáři je příklad seznamu *users.csv.example*, stačí jej tedy přejmenovat, případně editovat.

Verse server se pak spouští *./bin/verse_server*. Klient je spouštěn *./bin/verse_client adresa*. Zadáním *./bin/verse_client -h* je vypsána nápověda.